TOWARDS LEAN-GROUNDED MATHEMATICAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Formal verification and informal mathematical reasoning in LLMs have largely evolved independently. Informal outcome-level supervision provides limited guidance on how rigorous proofs should be structured, while invoking proof assistants such as Lean at inference time incurs recurring formalization costs. To address this, we propose *Lean-grounded Learning*, which uses Lean as privileged training-time supervision to improve informal reasoning without requiring Lean at inference. A teacher first generates an informal proof, then revises it using a verified Lean proof. A student is subsequently trained on these Lean-grounded revisions via SFT. On Qwen3-30B-A3B, Lean grounding improves AIME24 accuracy from 45.0 to 55.0 and ProofNet from 72.1 to 80.9 over the baseline with natural-language privileged information. With only 30K Lean-grounded examples, Qwen2.5-Math-7B-Base surpasses Qwen2.5-Math-7B-Instruct on MATH, AMC, and Olympiad Math, despite the latter being trained on roughly 2.5M examples followed by RL. Experiments show that Lean-grounded models more often derive claims from established facts, whereas Qwen3-30B-A3B more often relies on candidate testing and extrapolation from concrete cases.

1 INTRODUCTION

Scalable formal verification based on Lean 4 (De Moura et al., 2015) has emerged as a promising paradigm for verifying solutions generated by large language models (LLMs) (Polu et al., 2022; Ren et al., 2025; Wang et al., 2025a; Hubert et al., 2026) to challenging mathematical problems (OpenAI, 2026a;b; Alpöge & Furman, 2026). By formalizing mathematical arguments in Lean and checking every proof step with kernel, Lean provides a rigorous mechanism for judging reasoning correctness under dependent type theory. Beyond certifying correctness, a verified Lean derivation also exposes the intermediate proof obligations and logical dependencies through which the final conclusion is established. Meanwhile, informal mathematical reasoning (Cobbe et al., 2021; Lightman et al., 2024; Luong et al., 2025; Shao et al., 2025) lacks a robust mechanism for evaluating whole-proof correctness, and therefore often relies on simpler proxies, such as final-answer matching and rubric-based judgments. Training LLMs for advanced informal mathematical reasoning consequently tends to select or reward reasoning trajectories based primarily on whether they lead to correct outcomes (Shao et al., 2024; Guo et al., 2025). However, these outcome-level signals provide limited supervision on how the final artifact, such as a proof, should be structured to support a rigorous chain of deductions. Recent approaches harness LLMs' informal reasoning by invoking Lean 4 autoformalizers and provers (Liu et al., 2025a; Ospanov et al., 2025; Wang et al., 2025b) to validate each step in the proof, but the rapidly growing computational cost limits scalability during large-scale rollout. This motivates us to ask: *Can verified Lean 4 proofs provide a training-time signal that grounds informal reasoning in formal rigor, without requiring the reasoning process itself to be formalized at inference time?*

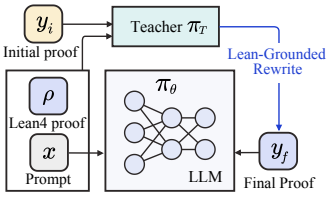

Figure 1: Lean-grounded learning.

Our empirical analysis suggests LLM-generated informal reasoning often fails to preserve the conditions under which intermediate claims are valid and the prior results on which subsequent deductions depend. Lean directly constrains both aspects: each local transformation must be verifiable under its current assumptions, while the dependencies supporting subsequent deductions remain explicit. We therefore propose *Lean-Grounded Learning* (overview in Figure 1), which leverages verified Lean derivations as privileged information for a teacher model capable of interpreting Lean

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

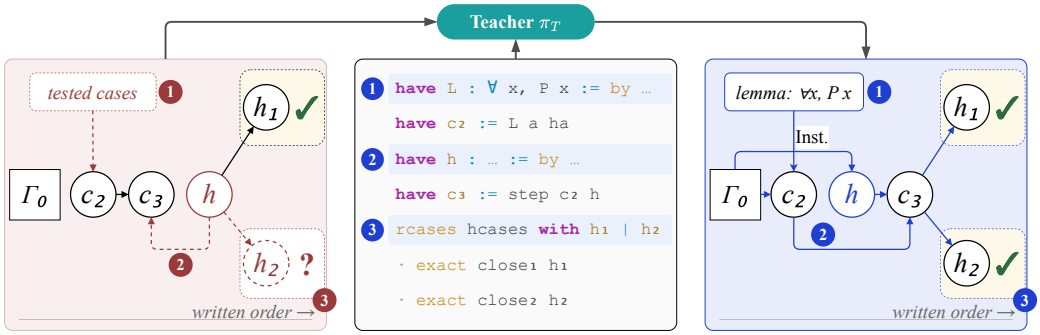

Figure 2: **Lean-grounded supervision transfers the structure of a verified Lean proof to an informal proof.** ① **Derivation:** in y_i, c_2 is supported only by tested cases (*e.g.*, Table 6). In ρ and y_f, c_2 follows from a proved lemma. ② **Dependency:** in y_i, c_3 uses condition h before h is established. In ρ and y_f, h is proved before it is used. ③ **Closure:** in y_i, the case h_2 is left open. In ρ and y_f, every case is closed.

proofs, thereby constructing offline supervision that transfers the structural inductive bias to informal mathematical reasoning via supervised fine-tuning (Ouyang et al., 2022). We design controlled experiments to isolate the effect of Lean grounding from the correctness of the privileged solution itself. Compared with supervision from the corresponding natural-language reasoning trajectory, Lean grounding yields consistent gains across both answer-seeking and proof-generation tasks. With Qwen3.6-35B-A3B as the teacher and Nemotron-Math-Proofs-v1 as the source dataset, Qwen3-8B improves from 23.3 to 31.7 pass@1 on AIME25 and from 26.7 to 32.9 on AIME26, while Qwen3-30B-A3B improves from 62.9 to 73.8 on FormalMATH-Lite and from 34.0 to 43.0 on CombiBench. In matched proof comparisons on IMO-ProofBench, Lean-Grounded is preferred in 60.8% of cases for derivation and 78.2% for dependency, compared with 9.2% and 4.6% for Qwen3-30B-A3B. The original Qwen3-30B-A3B more often tests candidates and extrapolates from concrete cases before establishing the underlying constraints, while Lean-grounded proofs more often derive claims from established facts. Beyond SFT, Lean-grounded initialization also leads to stronger RL performance on competition problems, reaching 20.0 on AIME25 and 16.7 on AIME26 after RL, compared with 10.0 on both benchmarks when RL starts from Qwen3-4B-Base. We further find that near-miss Lean proofs can still provide useful supervision, and that moving Lean verification to training can outperform test-time Lean assistance while avoiding its repeated formalization cost.

2 Lean-grounded Privileged Learning

2.1 A Constructive Proof-Theoretic View of Natural Proof Deduction

Inspired by the natural-deduction view of proofs (Gentzen, 1935; Prawitz, 1965), we describe a proof by the claims it asserts and the context in which each claim is asserted.

Definition 1 (Derivation and Dependency Discipline) *Let a proof be a sequence of claims $c_1, \ldots, c_n$, and let Γ_i denote the context after c_i: the hypotheses and branch conditions currently active, together with the facts derived under them. The context evolves through three operations:* derivation, $\Gamma_i = \Gamma_{i-1} \cup \{c_i\}$; extension, $\Gamma_i = \Gamma_{i-1} \cup \{h\}$, *when a hypothesis or branch condition h is introduced; and* discharge, *when h is resolved, which removes h together with every fact derived under it and retains only $h \Rightarrow c$, or c itself if c is derived in every branch of an exhaustive case split.*

The proof satisfies derivation discipline *if, for every i, $\Gamma_{i-1} \vdash c_i$ holds in a single justified step,* i.e., *by an inference rule, a computation, or a previously established lemma. In particular, no claim may be licensed by an unproven ansatz, by a pattern extrapolated from finitely many instances, or by numerical evidence. The proof satisfies* dependency discipline *if no claim depends on a hypothesis, or on a fact derived under it, after that hypothesis has been discharged; no hypothesis is dropped before it is discharged; and every transformation is applied only once its side conditions (e.g.,*

positivity, non-vanishing, domain membership) are in the context. Finally, the proof is closed *if no hypothesis remains active at c_n and every case split is exhaustive.*

A Lean 4 proof ρ satisfies all three conditions by construction: every tactic step is checked against the local context, hypotheses introduced by `intro`, `by_cases` or `rcases` remain scoped to their subgoals, lemmas stated with `have` must be proved before they are applied, and the proof compiles only once every goal is closed. Informal proofs enforce none of these conditions. Two further properties of ρ are relevant below. First, a Lean proof tends to decompose an argument into universally quantified lemmas that are instantiated explicitly (`have h : ∀x, ...; h a ha`), instead of invoking a global theorem on the whole expression. Second, the side conditions of each transformation appear as explicit hypotheses that must be supplied at the point of use. We therefore ask:

Research Question 1 (Transferability) *Can the deduction structure of Lean proofs transfer to LLMs' informal proof generation without access to Lean at inference time? If so, in what form does this transfer manifest in the resulting proofs?*

Research Question 2 (Emergent Abilities) *What effect does this transfer have on downstream mathematical reasoning, beyond the structural inductive bias itself?*

2.2 A Simple Implementation of Lean-Grounded Learning via SFT

To answer Research Question 1, we construct a supervision pipeline in which the verified derivation ρ is available only to the teacher as privileged information (Vapnik & Vashist, 2009). The teacher π_T first drafts an informal proof $y_i \sim \pi_T(\cdot \mid x)$ without access to ρ. It then revises the same draft using ρ, producing $y_f \sim \pi_T(\cdot \mid x, y_i, \rho)$. The revision prompt (Appendix A.1.2) asks the teacher to preserve correct content, repair only incorrect or missing reasoning, and never mention the exposure of Lean or any other artifact. The student π_θ is trained only on (x, y_f) pairs via SFT. We filter revisions that leak meta-information, such as "according to the Lean code," or differ substantially in length from y_i. At inference time, π_θ generates informal proofs without Lean, proof search, formal tool calls, or a verifier in the loop. We discuss more advanced implementations in Section A.4.

2.3 Case Study: How Lean-Grounded Revision Restructures Teacher Proofs

Case study in Table 1 answers Research Question 1, Lean grounding requires every step of the informal proof to be licensed by what has already been established. Shortcuts that informal proofs tolerate, such as recalled results, premature transformations, and incomplete checks, are replaced by explicit derivations. Even for a correct proof, Lean favors elementary lemmas that are proved once and instantiated explicitly over appeals to global theorems (Figure 2). In case (a), the task is to prove $\prod_{\text{cyc}}(\frac{1}{a}-1) \geq (\frac{1}{S}-1)^3$ for $0 < a, b, c < \frac{1}{2}$ and $S = a+b+c$. The initial proof y_i takes $\ln$ of both sides before checking the sign of $\frac{1}{S}-1$. In ρ, `by_cases` splits the goal into two subgoals, each carrying the sign condition in its local context, so no sign-sensitive step can be applied before the sign is known. The revision inherits this order. It first establishes LHS > 0, then splits on S, and applies $\ln$ and Jensen only when $S < 1$. Since the revision keeps the Jensen argument of y_i rather than the termwise comparison in ρ, what transfers is the ordering of obligations (dependency). In case (b), the task is to prove $\sum_{m=k^2+1}^{(k+1)^2} \sqrt{m} < 2k^2 + 2k + \frac{3}{2}$ for $k \geq 1$. The initial proof recalls a sum–integral comparison in the wrong direction. In ρ, `Finset.sum_le_sum` bounds a sum only from a proved bound on every term, here the tangent line of the concave $\sqrt{x}$, and `field_simp; ring` and `sub_lt_self` then evaluate the sum and make the inequality strict. The revision follows the same route, bounding each term and summing the bounds (derivation). In case (c), the task is to find the maximum of a^2b^3 for $a, b > 0$ with $\frac{a}{a+1} + \frac{2b}{b+1} = 1$. The initial proof declares a global maximum after checking f' at only two points. In ρ, a maximum consists of two goals, a universal `bound` over all feasible points and an existential `attain` closed by an explicit witness, and `exact ⟨attain, bound⟩` succeeds only when both are proved. The revision accordingly proves the bound for every $a > 0$ by AM–GM and exhibits the witness $(1, \frac{1}{3})$ (closure). In case (d), the task is to prove $a^3+b^3+c^3 \geq a^2+b^2+c^2$ for $a, b, c > 0$ with $abc = 1$. The initial proof is correct but relies on a WLOG ordering and Chebyshev's inequality. In ρ, a single lemma $x^3+1 \geq x^2+x$ is proved once by factoring, instantiated at each variable, and combined by `linarith`. The revision proves this inequality for every $x > 0$ in natural language and applies it to a, b, and c, so each step can be checked on its own without an ordering assumption or an external theorem.

Proof w/o Lean y_i | **Verified Lean proof ρ** | **Revision y_f**

(a) Dependency: a step is used before its side condition is proved. Given $0 < a, b, c < \frac{1}{2}$ and $S = a + b + c$, prove $\prod_{\text{cyc}}(\frac{1}{a} - 1) \geq (\frac{1}{S} - 1)^3$.

1. Take ln of both sides: $\sum \ln(\frac{1}{a} - 1) \geq 3\ln(\frac{1}{S} - 1)$.
2. Jensen on $g(x) = \ln \frac{1-x}{x}$: $\frac{1}{3}\sum g(a) \geq g(\frac{S}{3}) = \ln(\frac{3}{S} - 1) \geq \ln(\frac{1}{S} - 1)$.
3. Only at the end: "if $S \geq 1$ the RHS is ≤ 0." But step 1 already used $\ln(\frac{1}{S} - 1)$, which is undefined for $S \geq 1$.

```
have ha :  1/a-1 > 0 := ...
  -- same for b, c
by_cases h :  S ≥ 1
· have :  (1/S-1)^3 ≤ 0 := ...
  positivity -- LHS>0≥RHS
· have :  1/a-1 > 1/S-1 := ...
  -- same for b, c
  nlinarith -- multiply them
```

1. Since $a < \frac{1}{2}$, each $\frac{1}{a} - 1 > 1$, so LHS > 0.
2. Case $S \geq 1$: RHS $= (\frac{1}{S} - 1)^3 \leq 0 <$ LHS.
3. Case $S < 1$: both sides are positive, so take ln. Jensen on the convex $g(x) = \ln \frac{1-x}{x}$ gives LHS$^{1/3} \geq \frac{3}{S} - 1 > \frac{1}{S} - 1 > 0$; cube both sides.

Why: in Lean, a sign-sensitive step such as taking a logarithm cannot be applied until the sign is proved. `by_cases` splits the goal by the sign of $\frac{1}{S} - 1$, and each branch carries its own hypothesis. The revision keeps the Jensen argument of y_i and moves it into the branch $S < 1$, where the logarithm is defined.

(b) Derivation:Derivation: y_i uses a wrong sum–integral estimate, y_f bounds each term and sums the bounds. For $k \geq 1$, prove $\sum_{m=k^2+1}^{(k+1)^2} \sqrt{m} < 2k^2 + 2k + \frac{3}{2}$.

1. Recall a sum–integral comparison: $\sum \sqrt{m} \leq \int_{k^2}^{(k+1)^2} \sqrt{x}\, dx = 2k^2 + 2k + \frac{2}{3}$.
2. For increasing $\sqrt{x}$ the comparison goes the other way, so this bound is false ($k = 1$: $5.15 > 4.67$).
3. Conclude the target from the false bound.

```
have h :  ∀ m ∈ [k^2+1, (k+1)^2],
  √m ≤ t(m) := (k+1) + (m-(k+1)^2)/(2(k+1))
calc ∑ √m ≤ ∑ t(m)
    := Finset.sum_le_sum h
  _ = 2k^2+2k+3/2 - 1/(2(k+1))
    := by field_simp; ring
  _ < 2k^2+2k+3/2
    := sub_lt_self _ (by positivity)
```

1. $\sqrt{x}$ is concave, so it lies below its tangent at $x = (k+1)^2$: $\sqrt{m} \leq (k+1) + \frac{m-(k+1)^2}{2(k+1)}$.
2. Add the $2k+1$ bounds: the right-hand sides sum to $2k^2 + 2k + \frac{3}{2} - \frac{1}{2(k+1)}$.
3. Hence $\sum \sqrt{m} \leq 2k^2+2k+\frac{3}{2} - \frac{1}{2(k+1)} < 2k^2 + 2k + \frac{3}{2}$.

Why: in Lean, a bound on a sum is obtained with `Finset.sum_le_sum`, which requires a proved bound for every term. A sum–integral comparison cannot be cited without checking its hypotheses, and here it points the wrong way. The revision therefore bounds each $\sqrt{m}$ by the tangent line of $\sqrt{x}$ and adds the bounds.

(c) Closure: a maximum needs a global bound and a witness. For $a, b > 0$ with $\frac{a}{a+1} + \frac{2b}{b+1} = 1$, find the maximum of a^2b^3.

1. The constraint gives $b = \frac{1}{2a+1}$, so $a^2b^3 = f(a) = \frac{a^2}{(2a+1)^3}$.
2. $f'(1) = 0$ and $f(1) = \frac{1}{27}$.
3. Check the sign of f' at $a = \frac{1}{2}$ and $a = 2$ only, then declare $\frac{1}{27}$ the global maximum.

```
have bound :  ∀ a b,
  C a b → a^2*b^3 ≤ 1/27
have attain :  ∃ a b,
  C a b ∧ a^2*b^3 = 1/27
  := ⟨1, 1/3, ...⟩
exact ⟨attain, bound⟩
```

1. $b = \frac{1}{2a+1}$, as in the proof.
2. Bound for every $a > 0$: AM–GM on $a, a, 1$ gives $(2a+1)^3 \geq 27a^2$, so $a^2b^3 \leq \frac{1}{27}$.
3. Witness: $(a, b) = (1, \frac{1}{3})$ satisfies the constraint and attains $\frac{1}{27}$. Hence the maximum is $\frac{1}{27}$.

Why: in Lean, the claim that $\frac{1}{27}$ is the maximum splits into two goals, a bound for every feasible (a, b) and a feasible point attaining it, where C encodes positivity and the constraint. Checking the sign of f' at two points proves neither goal. The revision proves the bound by AM–GM and gives the witness $(1, \frac{1}{3})$.

(d) Local certificate: y_i calls a correct Chebyshev, Lean prefers a short check for each variable over a general theorem. For $a, b, c > 0$ with $abc = 1$, prove $a^3 + b^3 + c^3 \geq a^2 + b^2 + c^2$.

1. Assume $a \geq b \geq c$.
2. Chebyshev: $\sum a^3 \geq \frac{1}{3}(\sum a)(\sum a^2)$.
3. AM–GM: $\sum a \geq 3$, hence $\sum a^3 \geq \sum a^2$.

```
have h :  ∀ x > 0,
  x^3+1 ≥ x^2+x
  -- (x-1)^2*(x+1) ≥ 0
have hs :  a+b+c ≥ 3 := ...
linarith [h a ha, h b hb,
  h c hc, hs]
```

1. For every $x > 0$: $x^3 + 1 - x^2 - x = (x-1)^2(x+1) \geq 0$.
2. Apply at a, b, c and add: $\sum a^3 \geq \sum a^2 + (\sum a - 3)$.
3. AM–GM with $abc = 1$ gives $\sum a \geq 3$, so $\sum a^3 \geq \sum a^2$.

Why: applying Chebyshev's inequality requires the ordering $a \geq b \geq c$, which in Lean needs a separate symmetry argument. ρ instead proves one lemma $x^3 + 1 \geq x^2 + x$ by factoring and applies it to a, b, and c. The revision follows this route, so every step can be checked.

Table 1: **Case study of Lean-grounded revision.** Cases (a)–(c) show that the verified Lean proof ρ fixes the teacher's errors in the use of side conditions (dependency), unjustified estimates (derivation), and incomplete extremum arguments (closure). Case (d) shows that even when the teacher is correct, ρ still restructures the proof, replacing a global theorem that relies on an ordering assumption with a locally proved lemma, yielding a self-contained argument whose every step is locally checkable.

3 Experiments and Results

3.1 Experimental Setup

Datasets and evaluation. We evaluate two forms of informal mathematical reasoning. (1) *Answer-seeking*: MATH-500, AIME24–26, and IMO-AnswerBench (Luong et al., 2025), where a response is correct if its final answer matches the ground truth under Math-Verify. (2) *Proof generation*: ProofNet (Azerbayev et al., 2023), FormalMATH-Lite (Yu et al., 2025b), IMO-ProofBench (Luong

Methods	MATH	AIME24	AIME25	AIME26	IMOAnswer
Qwen3-8B Variants					
Qwen3-8B	84.8 / 94.4	26.5 / 39.1	19.2 / 29.8	16.0 / 28.3	16.4 / 31.1
Baselines					
Fine-Proof	84.2 / 93.6	33.3 / 53.3	30.0 / 46.7	24.2 / 43.3	7.9 / 20.5
OpenR1	80.0 / 89.4	20.0 / 30.0	18.3 / 26.7	12.5 / 20.0	12.2 / 26.3
Retool	78.7 / 94.2	31.7 / 46.7	22.5 / 43.3	26.7 / 50.0	14.0 / 29.8
Ablations					
Teacher CoT	82.6 / 94.4	32.5 / 50.0	25.8 / 30.0	23.3 / 43.3	16.7 / 35.2
Teacher Self-Refine	81.1 / 94.2	24.2 / 43.3	22.5 / 40.0	22.5 / 40.0	13.5 / 30.9
Teacher Traj	85.2 / 94.4	38.3 / 50.0	23.3 / 36.7	26.7 / 47.6	13.8 / 31.4
Teacher Lean (failed)	**87.2** / 95.6	39.2 / 60.0	27.5 / 43.3	29.2 / 53.3	**17.0** / 34.2
Teacher Lean	87.1 / **96.6**	**40.8 / 60.0**	**31.7 / 53.3**	**32.9 / 56.7**	16.9 / **35.4**
Qwen3-30B-A3B Variants					
Qwen3-30B	88.4 / 95.0	27.5 / 36.7	20.0 / 30.0	23.5 / 40.2	20.3 / 40.8
Ablations					
Teacher CoT	92.2 / **97.6**	46.7 / 66.7	35.0 / 50.0	38.1 / 59.8	23.9 / 52.8
Teacher Traj	91.6 / 97.2	45.0 / 56.7	30.0 / 46.7	36.7 / 56.4	22.5 / 53.5
Teacher Lean	**93.0** / 97.0	**55.0 / 76.7**	**38.3 / 56.7**	**45.2 / 67.8**	**26.5 / 53.8**

Table 2: **Ablation on the source of SFT supervision across student architectures.** Models are trained on proof problems from Nemotron-Math-Proofs-v1. Results report unbiased estimated Pass@1 / Pass@4 accuracy.

et al., 2025), and CombiBench (Liu et al., 2025b), using the natural-language statements of each problem. Proofs are graded on a 0–7 scale by GPT-OSS-120B, given the problem, a reference solution, and problem-specific grading guidelines where available (prompt in Appendix A.1.1). A proof counts as correct if it receives 6/7 or 7/7. Reference solutions are available for all problems in ProofNet, FormalMATH-Lite, and IMO-ProofBench, and for 45% of the problems in CombiBench. For the remaining CombiBench problems, the judge grades against the general rubric alone, with the prompt stating that no reference is provided. Our main training data source is Nemotron-Math-Proofs-v1 (Du et al., 2025), in which natural-language problems are formalized as Lean 4 theorems and proved by Goedel-Prover (Lin et al., 2026). Each compiler-verified proof comes with the prover's natural-language reasoning trajectory. We randomly sampled 10K distinct, uncontaminated problems using a fixed seed. In the controlled experiments (Section 3.1), all supervision conditions are built from the same 10K problems with the same teacher, and differ only in the privileged information provided to the teacher during revision. We remove training problems that duplicate benchmark problems using fuzzy matching.

Models and implementation details. Main experiments use Qwen3-8B and Qwen3-30B-A3B as students, and Qwen3.6-35B-A3B as the teacher, all in non-thinking mode. We use a maximum generation length of 8,192 tokens, temperature 0.6, and top-p 0.95. Each student is trained for 3 epochs with a peak learning rate of 1×10^{-5}, batch size 64, a cosine schedule, and the Adam optimizer on 8×H100 GPUs. The non-thinking setting removes long CoT self-exploration and self-reflection from the student's behavior. We adopt this setting to investigate how the reasoning bias from Lean shapes the student's behavior in downstream mathematical reasoning (Research Question 2). To test generalization beyond this setting (post-trained students trained on Nemotron), we additionally train weaker students, Qwen2.5-Math-7B-Base, Llama3.2-3B-Instruct, and Llama3.1-8B-Instruct, on Numina-Lean with DeepSeek-V3 as the teacher.

3.2 Lean-Grounded Learning Enhances Mathematical Reasoning in Post-Trained Models

Setting: Controlling the privileged information for the teacher. 1. *Teacher CoT*: the teacher generates proofs directly, without privileged information. 2. *Teacher Self-Refine*: the teacher critiques its own draft for unestablished claims, incomplete case analyses, and logical gaps, then repairs it without external evidence. 3. *Teacher Traj-Guided*: the teacher revises its draft using the natural-language reasoning trajectory accompanying the verified Lean 4 proof in Nemotron, which exposes a solution path but not the formal proof structure. 4. *Teacher Lean-Grounded*: the teacher revises its draft using the verified Lean 4 derivation ρ. Settings 2–4 apply the same single revision pass to the same drafts and are thus compute-matched.

Methods	ProofNet	IMO-Pro	FMATH	Combi
Qwen3-8B	51.8 / 68.2	1.3 / 5.0	55.3 / 67.8	29.5 / 39.0
Teacher CoT	57.0 / 75.4	4.5 / 15.0	56.9 / 69.7	31.0 / 42.0
Teacher Traj	54.4 / 76.0	4.6 / 15.0	57.5 / 72.7	32.3 / 42.0
Teacher Lean	**61.4 / 80.7**	**7.1 / 28.3**	**63.5 / 79.3**	**33.5 / 45.0**

(a) Qwen3-8B

Methods	ProofNet	IMO-Pro	FMATH	Combi
Qwen3-30B	65.2 / 81.8	2.1 / 3.3	60.4 / 69.6	33.1 / 45.0
Teacher CoT	76.9 / 89.7	9.2 / 21.7	70.4 / 84.5	40.3 / 51.0
Teacher Traj	72.1 / 88.2	7.3 / 20.0	62.9 / 77.4	34.0 / 44.0
Teacher Lean	**80.9 / 91.8**	**11.0 / 25.0**	**73.8 / 86.2**	**43.0 / 54.0**

(b) Qwen3-30B-A3B-Instruct

Table 3: Proof generation on Qwen3-8B and Qwen3-30B-A3B. We report pass@1 / pass@8 on IMO-ProofBench and pass@1 / pass@4 for other benchmarks.

Lean-grounded SFT improves competition-level mathematical reasoning. Although our training problems are mostly theorem-proving problems, the gains transfer to competition-level answer-seeking tasks. Qwen3-8B is a strong starting point: it has undergone large-scale SFT, reinforcement learning with verifiable rewards, and on-policy distillation from stronger teachers, and scores 26.5, 19.2, and 16.0 pass@1 on AIME24, AIME25, and AIME26. Fine-tuning on 10K Lean-grounded proofs raises these to 40.8, 31.7, and 32.9 (Table 2). The gain does not come from the revision pass itself: Teacher Self-Refine, which revises the same drafts without privileged information, reaches only 24.2, 22.5, and 22.5, and Teacher Traj-Guided, which receives a natural-language solution path, reaches 38.3, 23.3, and 26.7. Existing SFT corpora transfer less consistently: Fine-Proof improves AIME but drops IMO-AnswerBench from 16.4 to 7.9. On Qwen3-30B-A3B, Teacher Lean-Grounded is again the strongest setting at pass@1 on all five benchmarks, raising AIME24 from 27.5 to 55.0 and AIME26 from 23.5 to 45.2 on no-think setting.

Stability under sampling budgets and problem difficulty. We examine whether the advantage of Lean grounding holds across test-time sampling budgets and problem difficulty (Figure 3). Stability across students, datasets, and teachers is studied in Section 3.4. For each problem q, we draw n samples, let $c(q)$ be the number of correct ones, and compute the unbiased estimator $\widehat{\text{pass@}k}(q) = 1 - \binom{n-c(q)}{k} \Big/ \binom{n}{k}$. On AIME26 with $n = 16$, Lean-grounded supervision outperforms all other conditions at every budget, and its margin over Traj-Guided widens from $+8.5$ at pass@1 to $+15.9$ at pass@14 (81.7 vs. 65.8). Per-problem paired 95% confidence intervals exclude zero from pass@1 to pass@8, indicating that the improvement is not due to sampling noise across problems. To measure difficulty, we take the base model's per-problem solve rate $s_A(q) = c(q)/n$, so that a smaller $s_A(q)$ indicates a harder problem. Figure 3b plots the change $s_B(q) - s_A(q)$ after fine-tuning against $s_A(q)$. The gain is not uniform across difficulty. It is near zero on problems the base model already solves reliably, where little headroom remains, peaks at intermediate s_A, and remains positive on the hardest problems. Lean-grounded supervision yields the largest gain across the entire difficulty range.

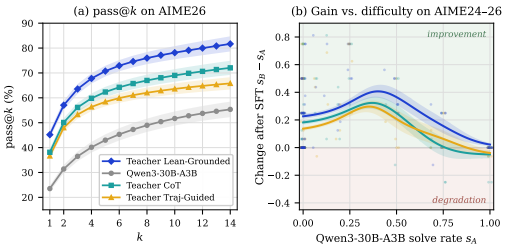

Figure 3: **Stability under sampling budget and problem difficulty (Qwen3-30B-A3B).** (a) pass@k on AIME26. (b) Per-problem gain against base solve rate s_A on AIME24–26.

3.3 Lean Grounding Transfers Proof Structure to the Student and Improves Proof Correctness in Post-Trained Models

Lean-grounded learning yields the highest proof correctness on all four proof benchmarks for both students (Table 3). On Qwen3-30B-A3B, pass@1 on ProofNet rises from 65.2 for the base model to 80.9 after Lean-grounded SFT, and on IMO-ProofBench from 2.1 to 11.0, compared with 72.1 and 7.3 after Traj-Guided SFT. On Qwen3-8B, pass@1 on ProofNet rises from 51.8 for the base model to 61.4 after Lean-grounded SFT, compared with 54.4 after Traj-Guided SFT. Both conditions expose the teacher to a solution path that leads to a plausibly correct proof, but only Lean grounding provides the formal derivation, in which intermediate goals, case splits, and dependencies are explicit. This difference shows up in how the student combines known facts. In Section A.2.2, all models derive the same bounds, but only the Lean-grounded student splits on S and applies each bound within the case where it holds, while the baselines chain two bounds in the same direction into false claims.

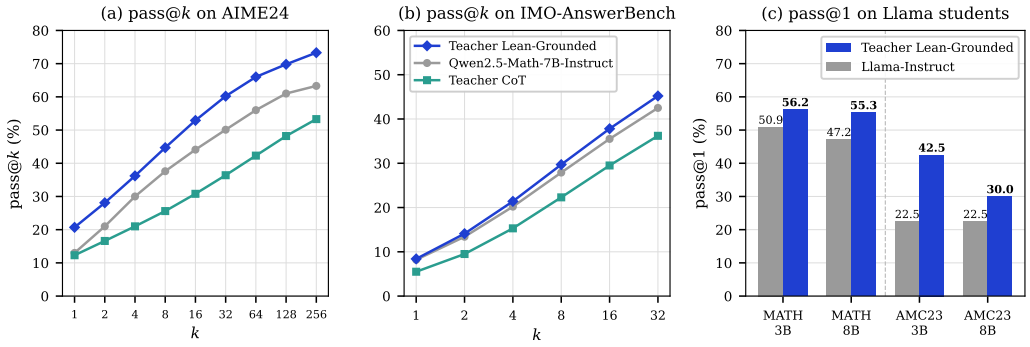

Figure 4: **Lean-grounded SFT on weaker and non-post-trained students.** (a,b) pass@k of Qwen2.5-Math-7B-Base fine-tuned on Teacher Lean-Grounded, compared with Qwen2.5-Math-7B-Instruct (trained on ∼2.5M examples plus RL), on AIME24 and IMO-AnswerBench. (c) pass@1 of Llama3.2-3B-Instruct and Llama3.1-8B-Instruct before and after Lean-grounded SFT on MATH-500 and AMC23.

Lean grounding shifts proof from exploratory testing to derivation from established facts. See Table 6, a recurring behavior of the base Qwen3-30B-A3B model is *exploratory testing* on IMO-ProofBench: the proof proposes candidate forms, enumerates small parameter values, or extrapolates a pattern from a few examples before establishing why the proposed structure must hold. After Lean-grounded learning, the decisive part of the argument is more often organized around direct deductions or a structural reformulation of the problem.

3.4 Generalize Lean-Grounded Learning to Weaker Students

Setting: Lean-grounded SFT on weaker and non-post-trained students. *Does the benefit depend on a student that is already strong at reasoning?* We train Qwen2.5-Math-7B-Base, Llama3.2-3B-Instruct, and Llama3.1-8B-Instruct. We use Numina-Lean and MATH-train. Numina-Lean provides compiled Lean proofs, we adopt DeepSeek-V3 as the teacher. For MATH-train, DeepSeek-V4-Flash-Preview performs autoformalization, proof generation, and Lean-grounded revision, and we keep only problems whose Lean proofs compile, yielding about 6K unique problems. For Qwen2.5-Math-7B-Base, we keep multiple revised proofs per problem, resulting in 27K training examples. We set the maximum generation length to 4,096 tokens, temperature to 0.6, and top-p to 0.95. All models are trained for 3 epochs. FLV (Cao et al., 2026) is a TIR baseline.

Lean-grounded SFT on a base model approaches or exceeds Qwen2.5-Math-7B-Instruct with about 1% of its data. Qwen2.5-Math-7B-Instruct is post-trained from the same base model on roughly 2.5M examples, followed by RL, and scores 83.3 on MATH and 60.0 on AMC23 (Table 4). With 27K Lean-grounded proofs for 6K MATH-train problems, the base model reaches 81.6 on MATH and 52.5 on AMC23, and exceeds the instruct model on CollegeMath (49.1 vs. 46.8) and OlympiadBench (43.3 vs. 41.6). With 30K Lean-grounded examples from Numina-Lean, it exceeds the instruct model on all four benchmarks, reaching 84.6 on MATH and 62.5 on AMC23. On the same 7K Numina problems with the same teacher, Lean-grounded supervision improves over Teacher CoT from 63.6 to 75.8 on MATH and from 29.3 to 39.1 on OlympiadBench. The same trend holds for Llama students (Figure 4): Lean-grounded SFT raises MATH accuracy from 50.9 to 56.2 on Llama3.2-3B-Instruct and from 47.2 to 55.3 on Llama3.1-8B-Instruct, and AMC23 accuracy from 22.5 to 44.5 and from 22.5 to 30.0, respectively.

Qwen2.5-7B	Budget	MATH	AMC23	Coll.	Olym.
Baselines					
Math-Base	0	44.0	27.5	20.7	16.3
Qwen-Instruct	∞	75.0	54.5	38.5	39.4
Qwen-Math-Instruct	2.5M	83.3	60.0	46.8	41.6
FLV	17K	78.6	30.6	48.0	16.6
MetaMATH	40K	51.2	27.5	34.7	16.7
OpenMath (MATH)	58K	74.4	47.5	47.1	36.4
Open R1	30K	76.2	60.0	38.8	34.1
Teacher CoT (Num.)	7K	63.6	45.0	30.1	29.3
Teacher Lean (MATH)	27K	81.6	52.5	49.1	43.3
Teacher Lean (Num.)	7K	75.8	57.5	40.2	39.1
Teacher Lean (Num. Fail)	7K	74.4	**62.5**	38.9	40.1
Teacher Lean (Num.)	30K	**84.6**	**62.5**	**49.3**	**47.3**

Table 4: **Lean-grounded SFT on Qwen2.5-Math-7B-Base.** Budget denotes the number of SFT examples. Teacher Lean (MATH) uses 27K proofs for 6K MATH-train problems. Teacher Lean (Num.) uses Numina-Lean problems. Qwen-Math-Instruct is additionally trained with RL.

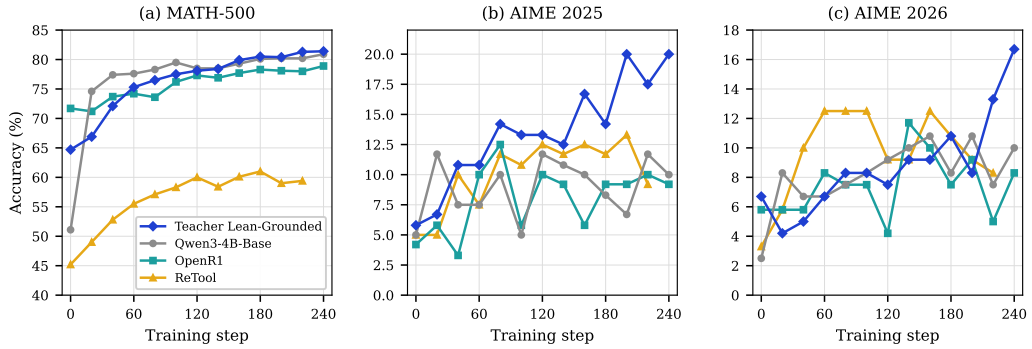

Figure 5: Lean-grounded SFT remains a stronger RL starting point at convergence for Qwen3-4B-Base with Math-Train Problems.

Figure 4 shows Lean-grounded learning on Qwen2.5-Math-Base outperforms the instruct model on AIME24 and IMO-Answerbench with larger Pass@k.

4 Discussions and Intriging Insights

4.1 Lean-grounded initialization improves RL on Qwen3-4B-Base.

We further ask *whether Lean-grounded SFT provides a better starting point for reinforcement learning* with settings in Section A.3.1. We show results in Figure 5 On MATH-500, all initializations except ReTool converge to a similar level, with Lean-grounded initialization reaching 81.4 at step 240, compared with 80.9 for the base model and 79.0 for OpenR1. The difference is larger on competition problems. At step 240, the Lean-grounded run reaches 20.0 on AIME 2025 and 16.7 on AIME 2026, compared with 10.0 and 10.0 for the base model and 9.2 and 8.3 for OpenR1. On AIME 2025, the Lean-grounded run leads from step 40 onward, whereas the other runs fluctuate around 10. This suggests that the proof structure acquired during Lean-grounded SFT is not overwritten by outcome-based RL, and instead provides a starting point from which RL continues to improve on harder problems.

Methods	IMO-Pro	AIME25
Qwen3-8B (thinking)	35.0	73.3
Teacher Lean (thinking)	**36.7**	**80.0**

Table 5: **Lean-grounded learning in thinking mode.** Here we report pass@4 on both IMO-ProofBench and AIME25.

4.2 Near-Miss Lean Proofs Can Still Provide Useful Supervision

We ask *whether a Lean proof that fails to compile can still provide useful supervision.* A proof that fails early or follows a wrong strategy exposes little of the structure a correct derivation would have. Nemotron-Math-Proofs-v1 is generated by Goedel-Prover, which revises its proof over multiple rounds with compiler feedback, and we take the proof from the first revision round, which already follows the structure of the final proof but still fails to compile. On Qwen3-8B, this supervision matches verified Lean grounding on MATH-500 and IMO-AnswerBench at Pass@1, trails it on AIME24–26, and exceeds Traj-Guided supervision on all five answer-seeking benchmarks (Table 2). Thus, A failed proof is useful when its structure is close to a valid derivation.

4.3 Lean-Grounded Learning in Thinking Mode: Preliminary Results

Main experiments use non-thinking mode to isolate the structural bias from ρ from the effects of long-CoT self-exploration and reflection. Extending Lean-grounded learning to long-context thinking mode is challenging as the teacher must selectively revise a long trace conditioned on ρ without disrupting valid exploratory steps such as testing and backtracking. We therefore apply GRAPE (Zhang et al., 2026) to keep the revised Nemotron-Math-Proof traces close to the student's own distribution. For each problem, we sample multiple Lean-grounded revisions and keep the one with the highest length-normalized likelihood under the student model. In Table 5, Lean-grounded SFT in thinking mode improves Qwen3-8B from 35.0 to 36.7 pass@4 on IMO-ProofBench and from 73.3 to 80.0 on AIME25 under 32K token length. Across the cases in Section A.2.3, longer thinking does

not stop the base model from licensing key claims by assumption, while the Lean-grounded student supplies a derivation or certificate for each of them, even on problems it does not fully solve.

4.4 Lean-Grounded Training vs. Test-Time Lean Assistance

Test-time methods such as HERMES (Ospanov et al., 2025) formalize and verify each reasoning step during inference. We run HERMES with its default decoding settings, using Qwen3-8B as the reasoner, Goedel-Formalizer-V2-8B and Goedel-Prover-V2-8B for each step, and a Lean REPL in the loop. On AIME25, it solves 4 of 30 problems (13.3 pass@1), below Qwen3-8B alone (19.2) and far below Lean-grounded SFT (31.7). HERMES spends 2,239 seconds and 15.5 Lean checks per problem on average. Most failures come from formalization. 47% of the 465 checks fail because the formalizer returns no compilable statement or the prover leaves the step as `sorry`. A passing check also does not guarantee a correct step, since the formal statement can be weaker than the claim. Section A.2.5 shows an example. On AIME25 I-3, HERMES verifies a correspondence between two sets that the formalizer defines identically, fails to formalize the actual correspondence, and gives up after 16 checks, while the Lean-grounded student solves the problem in 15 of 16 samples. Lean-grounded learning moves the cost of formal verification to training, where the derivation is used once to build supervision.

5 Related Works and Conclusion

Formal reasoning with Lean. Lean 4 (De Moura et al., 2015) provides a rigorous framework for verifying the mathematical reasoning of LLMs (Gulati et al., 2025; Yu et al., 2025b; Liu et al., 2025b; OpenAI, 2026a; Alpöge & Furman, 2026). Formal reasoning with Lean mainly involves: 1. autoformalization (Szegedy, 2020; Wu et al., 2022; Polu et al., 2023; Lu et al., 2024), which requires faithfully translating informal mathematical statements into formal Lean statements, and 2. formal theorem proving (Ren et al., 2025; Lin et al., 2025; Achim et al., 2025; Varambally et al., 2025; Wang et al., 2025a; Chervonyi et al., 2025), which LLMs construct a proof of a formal statement using tactics and lemmas.

Informal mathematical reasoning for LLMs. The informal mathematical reasoning capabilities of LLMs are commonly enhanced through: 1. SFT on large-scale verified, and long-CoT synthetic datasets generated by stronger teacher models (Yu et al., 2024; Toshniwal et al., 2024), and 2. online reinforcement learning with verifiable rewards (Shao et al., 2024; 2025), as well as logit distillation from teacher models (Lu & Lab, 2025; Kaur et al., 2026).

Bridging informal and Lean for LLMs. Prior work (Lin et al., 2024; Xin et al., 2024; Ren et al., 2025; Lin et al., 2026) leverages informal reasoners to construct Lean SFT datasets, using informal CoT for planning, lemma decomposition, self-verification, and exploration. At test time, Jiang et al. (2022); Varambally et al. (2025); Chung et al. (2026) use informal reasoners as a reasoning harness to enhance Lean 4 theorem proving. Conversely, Liu et al. (2025a); Ospanov et al. (2025); Wang et al. (2025b) explore using Lean 4 to verify intermediate steps in informal reasoning traces. In contrast, our work uses verified Lean 4 proofs (Du et al., 2025; Wang et al., 2025a) to improve informal mathematical reasoning at training time. Lean proofs serve only as privileged information for the teacher model when constructing SFT data, while the student performs reasoning without Lean tools at inference time. Compared with test-time verification, this approach avoids repeated autoformalization and proof-search costs for each query and provides step-level supervision on how a proof is derived, beyond what outcome-based rewards alone can provide. We further show that Lean-grounded SFT yields a stronger initialization for RL and that even unverified Lean proofs can provide useful supervision, suggesting that open-source Lean corpora can serve as a scalable source of training signal for informal reasoning.

Concluding Remarks. We propose *Lean-grounded learning*, which uses verified Lean 4 proofs as privileged information for teacher models to construct training data for student models. The resulting students generate structural proofs without access to Lean tools at inference time and outperform several supervision-based methods on natural-language solutions on both answer-seeking and proof-generation benchmarks in our settings. Our analysis suggests that these gains are accompanied by a shift in proof structure: students rely less on testing concrete cases and more on deriving claims from established facts and systematically closing cases. We hope this work motivates further research on integrating formal reasoning with informal CoT reasoning for training LLMs.

AI USE STATEMENT

We used commercial LLMs (Claude Opus 5.5 and ChatGPT 5.6) for two purposes: 1. to polish the language of this manuscript, and 2. to assist with qualitative analyses comparing the advantages of Lean-grounded proofs with other baselines. All model-assisted analyses were subsequently reviewed and verified by the authors. We take full responsibility for the final content. We also used open-source LLMs, including Qwen3.6, DeepSeek-V3, DeepSeek-V4-Flash-Preview, and Goedel-Prover, to synthesize the Lean-grounded dataset and construct baselines. Full details of their use are disclosed in the paper.

REPRODUCIBILITY STATEMENT

We provide the code for data construction, training, amd evaluation, at https://anonymous.4open.science/r/Lean_grounded_reasoning-7338. The repository includes the prompts used to construct Lean-grounded supervision and the corresponding baseline conditions, together with the scripts and configurations used for SFT and evaluation. The main experimental settings, including model choices, training hyperparameters, decoding parameters, dataset construction, and benchmark evaluation protocols, are described in Section 3.1 and Section A.

REFERENCES

Tudor Achim, Alex Best, Alberto Bietti, Kevin Der, Mathïs Fédérico, Sergei Gukov, Daniel Halpern-Leistner, Kirsten Henningsgard, Yury Kudryashov, Alexander Meiburg, et al. Aristotle: Imo-level automated theorem proving. *arXiv preprint arXiv:2510.01346*, 2025. 9

L Alpöge and R Furman. More than two thirds of the zeros of the riemann zeta function are simple and on the critical line. *arXiv preprint arXiv:2608.13637*, 2026. 1, 9

Zhangir Azerbayev, Bartosz Piotrowski, Hailey Schoelkopf, Edward W Ayers, Dragomir Radev, and Jeremy Avigad. Proofnet: Autoformalizing and formally proving undergraduate-level mathematics. *arXiv preprint arXiv:2302.12433*, 2023. 4

Chuxue Cao, Jinluan Yang, Haoran Li, Kunhao Pan, Zijian Zhao, Zhengyu Chen, Yuchen Tian, Lijun Wu, Conghui He, Sirui Han, et al. Pushing the boundaries of natural reasoning: Interleaved bonus from formal-logic verification. *arXiv preprint arXiv:2601.22642*, 2026. 7

Yuri Chervonyi, Trieu H Trinh, Miroslav Olšák, Xiaomeng Yang, Hoang H Nguyen, Marcelo Menegali, Junehyuk Jung, Junsu Kim, Vikas Verma, Quoc V Le, et al. Gold-medalist performance in solving olympiad geometry with alphageometry2. *Journal of Machine Learning Research*, 26 (241):1–39, 2025. 9

Jui-Hui Chung, Ziyang Cai, Zihao Li, Qishuo Yin, Rohit Agarwal, Simon Park, Rodrigo Porto, Narutatsu Ri, Ziran Yang, Shange Tang, et al. Goedel-architect: Streamlining formal theorem proving with blueprint generation and refinement. *arXiv preprint arXiv:2606.06468*, 2026. 9

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. 1

Leonardo De Moura, Soonho Kong, Jeremy Avigad, Floris Van Doorn, and Jakob von Raumer. The lean theorem prover (system description). In *International conference on automated deduction*, pp. 378–388. Springer, 2015. 1, 9

Wei Du, Shubham Toshniwal, Branislav Kisacanin, Sadegh Mahdavi, Ivan Moshkov, George Armstrong, Stephen Ge, Edgar Minasyan, Feng Chen, and Igor Gitman. Nemotron-math: Efficient long-context distillation of mathematical reasoning from multi-mode supervision. *arXiv preprint arXiv:2512.15489*, 2025. 5, 9

Gerhard Gentzen. Untersuchungen über das logische schließen. i. *Mathematische zeitschrift*, 39(1): 176–210, 1935. 2

Aryan Gulati, Brando Miranda, Eric Chen, Emily Xia, Kai Fronsdal, Bruno Dumont, Elyas Obbad, and Sanmi Koyejo. Putnam-axiom: A functional and static benchmark for measuring higher level mathematical reasoning in llms. *arXiv preprint arXiv:2508.08292*, 2025. 9

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. 1

Thomas Hubert, Rishi Mehta, Laurent Sartran, Miklós Z Horváth, Goran Žužić, Eric Wieser, Aja Huang, Julian Schrittwieser, Yannick Schroecker, Hussain Masoom, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651(8106):607–613, 2026. 1

Albert Q Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. *arXiv preprint arXiv:2210.12283*, 2022. 9

Simran Kaur, Narutatsu Ri, Yinghui He, Liam Fowl, and Sanjeev Arora. Rethinking on-policy self-distillation for thinking models. *arXiv preprint arXiv:2607.05184*, 2026. 9

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. 1, 21

Haohan Lin, Zhiqing Sun, Yiming Yang, and Sean Welleck. Lean-star: Learning to interleave thinking and proving. *arXiv preprint arXiv:2407.10040*, 2024. 9

Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, et al. Goedel-prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025. 9

Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, et al. Goedel-prover-v2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. In *International Conference on Learning Representations*, volume 2026, pp. 11793–11818, 2026. 5, 9

Chengwu Liu, Ye Yuan, Yichun Yin, Yan Xu, Xin Xu, Zaoyu Chen, Yasheng Wang, Lifeng Shang, Qun Liu, and Ming Zhang. Safe: Enhancing mathematical reasoning in large language models via retrospective step-aware formal verification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12171–12186, 2025a. 1, 9

Junqi Liu, Xiaohan Lin, Jonas Bayer, Yael Dillies, Weijie Jiang, Xiaodan Liang, Roman Soletskyi, Haiming Wang, Yunzhou Xie, Beibei Xiong, Zhengfeng Yang, Jujian Zhang, Lihong Zhi, Jia Li, and Zhengying Liu. Combibench: Benchmarking llm capability for combinatorial mathematics, 2025b. 5, 9

Jianqiao Lu, Yingjia Wan, Zhengying Liu, Yinya Huang, Jing Xiong, Chengwu Liu, Jianhao Shen, Hui Jin, Jipeng Zhang, Haiming Wang, et al. Process-driven autoformalization in lean 4. *arXiv preprint arXiv:2406.01940*, 2024. 9

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation. 9, 21

Minh-Thang Luong, Dawsen Hwang, Hoang H Nguyen, Golnaz Ghiasi, Yuri Chervonyi, Insuk Seo, Junsu Kim, Garrett Bingham, Jonathan Lee, Swaroop Mishra, et al. Towards robust mathematical reasoning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 35406–35430, 2025. 1, 4, 14

Sadegh Mahdavi, Branislav Kisacanin, Shubham Toshniwal, Wei Du, Ivan Moshkov, George Armstrong, Renjie Liao, Christos Thrampoulidis, and Igor Gitman. Scaling generative verifiers for natural language mathematical proof verification and selection. *arXiv preprint arXiv:2511.13027*, 2025. 14

OpenAI. Finite time blowup for navier–stokes. https://cdn.openai.com/pdf/32d9f210-8b73-45e0-91bc-82a30aef8a9a/navier-stokes.pdf, September 2026a. Manuscript. 1, 9

OpenAI. Ten advances in mathematics and theoretical computer science. https://cdn.openai.com/pdf/ten-proofs-oai.pdf, August 2026b. Updated August 6, 2026. 1

Azim Ospanov, Zijin Feng, Jiacheng Sun, Haoli Bai, Xin Shen, and Farzan Farnia. Hermes: Towards efficient and verifiable mathematical reasoning in llms. *arXiv preprint arXiv:2511.18760*, 2025. 1, 9

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022. 2

Ivo Petrov, Jasper Dekoninck, Dimitar I. Dimitrov, and Martin Vechev. Not all proofs are equal: Evaluating llm proof quality beyond correctness, 2026. URL https://arxiv.org/abs/2605.10379. 14

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022. 1

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=-P7G-8dmSh4. 9

Dag Prawitz. *Natural Deduction: A Proof-Theoretical Study*. Almqvist & Wiksell, Stockholm, 1965. 2

ZZ Ren, Zhihong Shao, Junxiao Song, Huajian Xin, Haocheng Wang, Wanjia Zhao, Liyue Zhang, Zhe Fu, Qihao Zhu, Dejian Yang, et al. Deepseek-prover-v2: Advancing formal mathematical reasoning via reinforcement learning for subgoal decomposition. *arXiv preprint arXiv:2504.21801*, 2025. 1, 9

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Y Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 1, 9

Zhihong Shao, Yuxiang Luo, Chengda Lu, ZZ Ren, Jiewen Hu, Tian Ye, Zhibin Gou, Shirong Ma, and Xiaokang Zhang. Deepseekmath-v2: Towards self-verifiable mathematical reasoning. *arXiv preprint arXiv:2511.22570*, 2025. 1, 9

Christian Szegedy. A promising path towards autoformalization and general artificial intelligence. In *International Conference on Intelligent Computer Mathematics*, 2020. 9

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024. 9

Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 22(5-6):544–557, 2009. 3

Sumanth Varambally, Thomas Voice, Yanchao Sun, Zhifeng Chen, Rose Yu, and Ke Ye. Hilbert: Recursively building formal proofs with informal reasoning. *arXiv preprint arXiv:2509.22819*, 2025. 9

Haiming Wang, Mert Unsal, Xiaohan Lin, Mantas Baksys, Junqi Liu, Marco Dos Santos, Flood Sung, Marina Vinyes, Zhenzhe Ying, Zekai Zhu, et al. Kimina-prover preview: Towards large formal reasoning models with reinforcement learning. *arXiv preprint arXiv:2504.11354*, 2025a. 1, 9

Ruida Wang, Yuxin Li, Yi R Fung, and Tong Zhang. Let's reason formally: Natural-formal hybrid reasoning enhances llm's math capability. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 16794–16820, 2025b. 1, 9

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *NeurIPS*, 2022. 9

Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152*, 2024. 9

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *ICLR*, 2024. 9

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale, 2025. *URL https://arxiv. org/abs/2503.14476*, 1(2), 2025a. 21

Zhouliang Yu, Ruotian Peng, Keyi Ding, Yizhe Li, Zhongyuan Peng, Minghao Liu, Yifan Zhang, Zheng Yuan, Huajian Xin, Wenhao Huang, et al. Formalmath: Benchmarking formal mathematical reasoning of large language models. *arXiv preprint arXiv:2505.02735*, 2025b. 4, 9

Dylan Zhang, Qirun Dai, and Hao Peng. The best instruction-tuning data are those that fit. *Advances in Neural Information Processing Systems*, 38:141172–141208, 2026. 8

A Appendix

A.1 Prompts

A.1.1 GPT-OSS Evaluation for Proof Correctness.

We use GPT-OSS-120B as an automatic judge for proof correctness for three reasons. First, final-answer matching is insufficient for proof-generation tasks: a proof can reach the correct final claim while containing an invalid derivation, so grading must be based on more fine-grained consideration of the proof, which requires a judge capable of following multi-step mathematical reasoning. Second, prior work has shown that LLM-based judges, when given a reference solution and explicit grading guidelines, achieve high agreement with expert human graders on this exact task, and high similar agreement is observed by Mahdavi et al. (2025) and Petrov et al. (2026), indicating that a sufficiently capable LLM judge, properly grounded in a reference solution, can approximate expert grading at a fraction of the cost. Third, grading at our scale is infeasible for human experts to perform consistently, and a fixed automatic judge additionally removes inter-rater variance across our many ablation conditions, so that comparisons between conditions are not confounded by grader drift. We use the prompt shown in Section A.1.1, adapted from Luong et al. (2025), which requires the judge to compare a proposed solution against a ground-truth reference and problem-specific grading guidelines, following the same design that prior work found necessary for high agreement with human graders.

Prompt for GPT-OSS-120B to Judge Proof Correctness

You are an expert grader for the International Mathematics Olympiad (IMO). Your task is to evaluate a proposed solution strictly and rigorously. Keep in mind the standards at the IMO are extremely high: only arguments that are logically sound, complete, and precise should be rewarded.

General Scoring Rubric

Scores are assigned on a 0-7 scale. The general guidelines are:

7 Points (Correct): The solution is complete, correct, and fully rigorous. If the submission contains incorrect attempts or lines of reasoning but ultimately presents a complete and correct solution, it should still be awarded full points; the presence of earlier, discarded work does not detract from the final correct proof.

6 Points (Almost Correct): The solution is almost correct with a sound core argument, but contains minor errors in calculation or small gaps in logic. Missing proofs for major components, unjustified claims, or sketchy arguments are **not** eligible for 6 points.

1 Point (Partial Progress): The solution demonstrates substantial progress explicitly mentioned in the grading guidelines. Initial observations, reformulating the problem without making substantive headway, or proving partial results not mentioned in the grading guidelines are generally **not** eligible for this score.

0 Points (Incorrect): The solution doesn't make substantial progress that is a key step in the full solution or is fundamentally flawed. All partial progress without key results or lacking rigor also fall in this category.

Input Data and Interpretation

You are provided with the following:

1. **Problem Statement:** The IMO problem.
2. **Ground Truth Solution:** A reference solution. Assume this solution is correct. It demonstrates one valid approach.
3. **Specific Grading Guidelines:** Criteria for awarding credit for this specific problem. These guidelines take precedence over the General Scoring Rubric, especially for partial credit.
4. **Proposed Solution:** The student submission.

Evaluation Process

You must follow this structured process:

1. **Analyze References:** Meticulously read and understand the problem and Ground Truth Solution check the Specific Grading Guidelines. Identify the key steps for a complete solution and the criteria for partial credit.
2. **Step-by-Step Verification:** Verify the logical validity and rigor of every step. Identify all flaws, gaps, assumptions, and errors. **Make sure you fully understand every piece of logic behind each step of the proposed solution, you must be careful for solutions that 'pretend' to be correct.**
3. **Assess Progress:** Determine the extent of non-trivial progress made.
4. **Score Determination:** Compare the findings against the Specific Grading Guidelines and the General Rubric to determine the final score.

Output Requirements

You must provide your final score in the format `<points>N out of 7</points>`. Ensure the `<points>` block is used **only once**, as your answer will be parsed based on the first `<points> </points>` block that appears in your whole response.

PROBLEM STATEMENT

{problem_statement}

GROUND-TRUTH SOLUTION

{solution}

SPECIFIC GRADING GUIDELINES

{guidelines}

PROPOSED SOLUTION

{student_answer}

Present your detailed thought process and formal justification based on the scoring rubric and grading guidelines, and finally present your final score in the format below.

[Select one of the following options]
<points>7 out of 7</points>
<points>6 out of 7</points>
<points>1 out of 7</points>
<points>0 out of 7</points>

For problems without specific grading guidelines, we treat the prompt as a general rubric for all baselines for fair comparison.

A.1.2 Prompts for Lean-Grounded Proof Refinement

Prompt for Lean-Grounded Proof Refinement

```
You are an expert mathematician.
You will receive a potentially incorrect natural-language mathematical proof
and a verified Lean 4 proof. Conservatively repair the natural-language
reasoning using the Lean proof as private evidence to make the proof correct.

The curated proof should:
1. preserve all relevant correct mathematical content, explanations,
   equations, derivations, and the original final answer;
2. edit only incorrect or missing reasoning needed for a sound proof;
3. preserve the order, section headings, numbered steps, intermediate
   justifications, and step-by-step structure of the source proof;
4. retain at least the same explanatory depth and approximately the same
   length. Do not summarize, compress, merge steps, or replace a derivation
   with a terse citation. Remove material only when it is mathematically wrong
   or directly misleading. The repaired proof may be longer when needed;
5. be self-contained natural mathematical prose;
6. never mention Lean, formal verification, tactics, curation, hidden evidence,
   or any original/reference/provided proof, internal analysis, formalization,
   implementation, or source layer. Ordinary discussion of an explicit user
   request or of the visible problem's requested answer format is allowed and
   is not private-process leakage.
7. Preserve detailed or verbose reasoning from the original response whenever
   it contributes to the derivation or verification. Do not shorten correct
   reasoning merely as a matter of style.

The original natural language problem:
{problem}

The original natural language proof:
{proof}

The verified Lean proof for reference only:
{lean}

Return only the curated proof within <proof></proof>.
```

A.1.3 Prompts for Teacher Self-Refine

Prompt for Teacher's Self-Refinement

```
Below is a problem and a candidate proof. Your task is to critically review
the proof and then produce an improved version.

Problem:
{problem_statement}

Candidate proof:
{initial_proof}

First, critique the candidate proof. Examine specifically:
1. Are there claims used without being established first?
2. Are any case analyses incomplete or missing?
3. Are there logical gaps where a step does not follow from what precedes it?
4. Are any dependencies between intermediate results left implicit?
5. Is any step's justification vague or hand-waved?

Then write a revised proof that addresses every issue you identified. The
revised proof must be self-contained and follow the same requirements as
before: explicit intermediate claims, explicit case analyses, justified
transitions, natural language only.

Output format:
<critique>
...your critique...
</critique>
<revised_proof>
...your revised proof...
</revised_proof>
```

Problem Statement	Qwen3-30B-A3B	Lean-Grounded
PB-Basic-003 — Find all functions $f\colon \mathbb{R} \to \mathbb{R}$ such that for all $a, b \in \mathbb{R}$, $(b-a)f(f(a)) = af(a+f(b))$.	**Candidate-form testing.** Tests $f = 0$, constant functions $f(x) = c$, and linear functions $f(x) = kx$, then attempts to exclude nonlinear alternatives.	**Identity-driven deduction.** Specializes the equation at $b = a$, $a = 0$, and $b = 0$ to derive reusable identities such as $f(f(0)) = 0$ and $f(a + f(a)) = 0$, using these constraints to narrow the solution space before determining the final form.
PB-Basic-004 — Find all strictly increasing functions $g : \mathbb{R} \to \mathbb{R}$ such that: (a) g is surjective; (b) $g(g(x)) = g(x) + 20x$.	**Affine ansatz.** Assumes $g(x) = ax + b$, obtaining $g(x) = 5x$ and $g(x) = -4x$, and rejects the latter using monotonicity.	**Monotonicity-constrained deduction.** First derives $g(0) = 0$ and sign preservation from monotonicity and surjectivity, then combines these structural constraints with the iteration equation to isolate $g(x) = 5x$.
PB-Basic-005 — Let P be a polynomial with real coefficients whose leading coefficient is 1. Suppose that for all nonzero real numbers x, $P(x) + P(1/x) = \frac{P(x+1/x)+P(x-1/x)}{2}$. Determine all possibilities for P.	**Low-degree experimentation.** Tests $P(x) = x$, x^2, x^4, $x^2 + a$, and a general quadratic to infer the likely form.	**Coefficient-level constraints.** Recasts the equation as a Laurent-polynomial identity and compares coefficients, turning observations about particular low-degree examples into explicit algebraic constraints on the degree and coefficients of P.
PB-Basic-007 — Determine all positive integers n and integer sequences $a_0, a_1, \ldots, a_n$ such that $a_n = 3$ and $f(a_{i-1}) = a_i$ for all $i = 1, 2, \ldots, n$, where $f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$.	**Small-case enumeration.** Checks $n = 1, 2, 3$ and tries concrete coefficient values, then extrapolates uniqueness from the failed cases.	**Divisibility-guided reduction.** Uses $f(a_{n-1}) = 3$ to obtain an integer-root divisibility condition $a_{n-1} \mid (a_0 - 3)$, and propagates this constraint together with $f(a_{i-1}) = a_i$ to restrict the possible sequences.
PB-Basic-020 — Find all pairs of primes (a, b) such that $a^2 - ab - b^3 = 1$.	**Parameter search.** Tries $b = 2, 3, 5, 7, 11$, and then $a = 2, 3, 5, 7, 11, 13$, identifying $(7, 3)$ from these trials.	**Factorization and prime divisibility.** Transforms the equation into $a(a-b) = (b+1)(b^2 - b + 1)$ and uses primality together with $\gcd(b+1, b^2-b+1) = \gcd(b+1, 3)$ to reduce the search to a small number of forced divisibility cases.
PB-Basic-023 — Suppose a, b, c are positive integers such that $2^a + 1 = 7^b + 2^c$. Find all possible values of (a, b, c).	**Case-by-case numerical search.** Tests $b = 1, 2, 3, 4$ and multiple values of c, continuing to rely on concrete cases even after factoring the equation.	**Congruence and valuation constraints.** Modulo 7, derives $a \equiv 0 \pmod 3$ and $c \equiv 1 \pmod 3$, then studies $2^c(2^{a-c} - 1) = 7^b - 1$ through parity and powers of 2, sharply restricting the admissible cases before solving them.
PB-Basic-024 — Find all non-negative integers a, b, c such that $20^a + b^4 = 2024^c$.	**Numerical enumeration.** Checks $c = 0, 1, 2$ and several values of a, repeatedly testing whether the remainder is a fourth power.	**Modular elimination.** Organizes the proof by cases in c and uses congruences modulo $16, 32, 64$, and 128 to eliminate entire families of possibilities rather than relying only on individual numerical checks.
PB-Advanced-014 — Two distinct positive integers a and b are written on a blackboard. At each step, you must choose one of the two operations, either adding 2 or multiplying by 3, for each number separately, and apply the chosen operation to both numbers (possibly using different operations). You must apply an operation to both numbers every step. Find all pairs (a, b) such that we can make the two numbers on the blackboard equal after a finite number of steps.	**Pattern exploration.** Simulates pairs such as $(1, 2)$, $(1, 3)$, and $(1, 4)$, tracking their differences to search for a pattern.	**Invariant-based reasoning.** Shows that the parity of the difference is preserved under all four possible transitions, then refines this invariant modulo 4 to characterize which initial residue classes can reach equality.
PB-Advanced-012 — Let p be a prime number and a, b be positive integers such that $p^n = a^4 + b^4$ for some integer $n \geq 2$. Prove that $n \geq 5$.	**Testing small exponents.** For $n = 2, 3, 4$, checks concrete pairs such as $(1, 1)$, $(1, 2)$, $(2, 2)$, and $(2, 3)$, and extrapolates impossibility from the observed values.	**GCD and residue reduction.** Factors out $g = \gcd(a, b)$ to reduce to a coprime equation, then uses fourth-power residues modulo 5 and 16 to constrain the prime power structurally and rule out $n = 2, 3, 4$.

Table 6: **Lean grounding shifts proof generation from exploratory testing toward structure-guided deduction.** Qwen3-30B-A3B often advances by testing candidate forms, enumerating concrete parameter values, or extrapolating patterns from examples. Lean-Grounded proofs instead more often derive reusable identities, divisibility conditions, congruence constraints, or invariants that restrict the subsequent proof and reduce the need for unguided exploration.

A.2 Additional Case Study

A.2.1 Lean grounding shifts proof from exploratory testing to derivation from established facts

See examples in Table 6.

A.2.2 LEAN-GROUNDED LEARNING TEACHES STUDENTS TO APPLY KNOWN FACTS ONLY WHERE THEY HOLD

See cases below.

Case study: combining two bounds in the same direction (IMO-ProofBench, PB-Basic-002)

Problem. Show that $x^2+y^2+z^2+t^2 \geq xyzt$ for any positive real numbers x, y, z, t that satisfy $2(x+y+z+t) \geq xyzt$.

Qwen3-30B-A3B

We know from the AM-GM inequality that [...] $x+y+z+t \geq 4\sqrt[4]{xyzt}$. So: $2(x+y+z+t) \geq 8\sqrt[4]{xyzt}$. But we are given that: $2(x+y+z+t) \geq xyzt$. So combining these: $xyzt \leq 2(x+y+z+t) \geq 8\sqrt[4]{xyzt}$. This implies: $xyzt \leq 8\sqrt[4]{xyzt}$. [...] Therefore: $\sqrt[4]{xyzt} \leq 2 \Rightarrow xyzt \leq 16$. So we have shown that $xyzt \leq 16$. [...] Divide both sides by $xyzt$: $xyzt \geq 16$. But earlier, we showed that $xyzt \leq 16$, so the only way this can be true is if: $xyzt = 16$. [...] This completes the proof.

✗ Both $2(x+y+z+t) \geq xyzt$ and $x+y+z+t \geq 4\sqrt[4]{xyzt}$ are *lower* bounds on the same sum, so they cannot be chained; the claim $xyzt \leq 16$ is false.

Teacher CoT

We know $2(x+y+z+t) \geq xyzt$. By the AM-GM inequality applied to the linear terms x, y, z, t: $x+y+z+t \geq 4\sqrt[4]{xyzt}$. Substitute this into the given constraint $2(x+y+z+t) \geq xyzt$: $2(4\sqrt[4]{xyzt}) \geq xyzt$, $8\sqrt[4]{xyzt} \geq xyzt$. [...] $u \leq 2$, $\sqrt[4]{xyzt} \leq 2$, $xyzt \leq 16$. So, the condition $2(x+y+z+t) \geq xyzt$ implies that the product $xyzt$ is bounded above by 16. [...] Therefore, $x^2+y^2+z^2+t^2 \geq 4\sqrt{xyzt} \geq xyzt$.

✗ The same invalid substitution: a lower bound on $x+y+z+t$ is substituted into another lower bound on it.

Teacher Traj-Guided

Let $S = x+y+z+t$ and $P = xyzt$. [...] From AM-GM on x, y, z, t: $S \geq 4\sqrt[4]{P} \implies \sqrt[4]{P} \leq \frac{S}{4}$, $P \leq \frac{S^4}{256}$. We also know $2S \geq P$. Substituting the upper bound for P: $2S \geq \frac{S^4}{256}$, $512S \geq S^4$, $512 \geq S^3$, $S \leq \sqrt[3]{512} = 8$. So, the sum S is bounded by $S \leq 8$. [...] $P \leq \frac{8^4}{256} = 16$. Therefore, $P \leq 16$ is indeed true for all positive real numbers satisfying the given condition.

✗ $P \leq 2S$ and $P \leq \frac{S^4}{256}$ are both *upper* bounds on P, so they imply nothing about $2S$ versus $\frac{S^4}{256}$; the claim $S \leq 8$ is false.

Teacher Lean-Grounded

Let $S = x+y+z+t$ and $P = xyzt$. The condition becomes $S \geq \frac{P}{2}$, or equivalently $P \leq 2S$. [...] Let $Q = x^2+y^2+z^2+t^2$. [...] **Case 1:** $S \leq 8$**.** 1. By Cauchy-Schwarz (or QM-AM), $Q \geq \frac{S^2}{4}$. 2. By AM-GM, $P \leq (\frac{S}{4})^4 = \frac{S^4}{256}$. 3. We check if $\frac{S^2}{4} \geq \frac{S^4}{256}$: $64 \geq S^2 \implies S \leq 8$. Since we are in the case $S \leq 8$, this holds. 4. Therefore, $Q \geq \frac{S^2}{4} \geq \frac{S^4}{256} \geq P$. So, $Q \geq P$. **Case 2:** $S > 8$**.** 1. By Cauchy-Schwarz, $Q \geq \frac{S^2}{4}$. 2. By the given condition, $P \leq 2S$. 3. We check if $\frac{S^2}{4} \geq 2S$: $\frac{S^2}{4} \geq 2S \iff S \geq 8$. Since we are in the case $S > 8$, this holds. 4. Therefore, $Q \geq \frac{S^2}{4} \geq 2S \geq P$. So, $Q \geq P$. In all cases, the inequality $x^2+y^2+z^2+t^2 \geq xyzt$ holds.

✓ Uses the same two upper bounds as Traj-Guided, but splits on S and applies each bound only in the case where it is the tighter one.

Why: in Lean, $P \leq 2S$ and $P \leq \frac{S^4}{256}$ cannot be combined into a bound on S; using both requires a case split such as `rcases le_or_lt S 8`, which opens one goal per case with the case hypothesis in context. The false intermediate claims fail at $(x, y, z, t) = (1, 1, 1, 20)$, which satisfies the condition with $S = 23$ and $P = 20$.

A.2.3 CASES ON THINKING MODE

Case study in thinking mode: deriving the form of f (IMO-ProofBench, PB-Basic-001)

Problem. Determine all functions $f : \mathbb{Z} \to \mathbb{Z}$ such that, for all $x, y \in \mathbb{Z}$, we have $f(2x) + 2f(y) = f(f(x+y))$.

Base model (thinking mode)

From Equation 1 and Equation 2, we can derive: $f(f(x)) = 2f(x) + f(0)$. [...] Let us now suppose that f is linear, i.e., of the form: $f(x) = ax + b$. [...] Thus, the only valid linear solutions are: $f(x) = 0$ (when $a = 0$, $b = 0$), $f(x) = 2x + b$ (when $a = 2$, any $b \in \mathbb{Z}$). [...] Let us now consider whether other functions (non-linear) can satisfy the equation. [...] If f is not surjective, then the function may not be fully determined by this relation. However, through recursive application of Equation 2' and Equation 1', we can show that any function satisfying the original equation must be of the form: $f(x) = 2x + c$ for some $c \in \mathbb{Z}$ or the zero function.

✗ The linear form is assumed, and the claim that no non-linear solution exists is asserted without proof.

Lean-Grounded (thinking mode)

Let $c = f(0)$. [...] $c + 2f(y) = f(f(y))$. (2) Substitute (2) into (1): $f(2x) = (c + 2f(x)) - 2c = 2f(x) - c$. (3) Using $P(x, y)$ and substituting $f(2x)$ from (3) and $f(f(x+y))$ from (2) with argument $x+y$: $(2f(x)-c)+2f(y) = 2f(x+y)+c$. Rearranging terms: $2f(x) + 2f(y) - 2c = 2f(x + y) \implies f(x + y) = f(x) + f(y) - c$. (4) Define a new function $g(x) = f(x) - c$. Substituting $f(x) = g(x) + c$ into (4): [...] $g(x + y) = g(x) + g(y)$. Since $g : \mathbb{Z} \to \mathbb{Z}$ is additive, it is linear. Specifically, for any $n \in \mathbb{Z}$, $g(n) = n \cdot g(1)$. [...] Consequently, $f(x) = g(x) + c = kx + c$. [...] **Case 1:** $k = 0$. [...] This yields the solution $f(x) = 0$. **Case 2:** $k = 2$. [...] This yields the family of solutions $f(x) = 2x + c$ for any $c \in \mathbb{Z}$.

✓ Derives Cauchy's equation from the hypothesis, so the linear form is proved before it is used, and both cases are kept.

Why: a Lean proof cannot posit $f(x) = ax + b$. The form of f must be derived from the hypothesis, and the Lean-grounded student does so through Cauchy's equation on $\mathbb{Z}$. Valid proofs on this problem: base 0%, Lean-grounded 75%.

Case study in thinking mode: bounding a count from both sides (IMO-ProofBench, PB-Basic-014)

Problem. In Shoreline Amphitheatre, 2024 Googlers each hold up to five flags of various colors. Any group of three Googlers will always have at least two people holding flags of the same color. Prove that one specific flag color is held by at least 200 Googlers.

Base model (thinking mode)

[...] So, $C \geq 50$. [...] Now, the total number of color assignments is: $\sum_{i=1}^{C} n_i \leq 5n = 10120$. Since there are at least 50 colors, the average number of Googlers per color is: $\frac{10120}{C} \leq \frac{10120}{50} = 202.4$. By the pigeonhole principle, at least one color must be held by at least: $\lceil 202.4 \rceil = 203$ Googlers.

✗ An upper bound on the average gives no lower bound on any single color, so the pigeonhole step is invalid.

Lean-Grounded (thinking mode)

We model the situation using a graph $G = (V, E)$ [...] An edge $\{u, v\}$ exists in E if and only if Googler u and Googler v share at least one common flag color. [...] Therefore, the graph G has no independent set of size 3 [...] Equivalently, the complement graph $\bar{G}$ contains no triangles (K_3). [...] $|E(G)| = \binom{n}{2} - |E(\bar{G})| \geq \binom{n}{2} - \frac{n^2}{4}$. [...] Suppose, for the sake of contradiction, that every color is held by at most 199 Googlers. [...] Since each Googler is in at most 5 cliques, and each $|C_c| \leq 199$, the number of edges incident to u is at most $5 \times 198 = 990$. Summing over all n Googlers and dividing by 2 (since each edge is counted twice), we get: $|E(G)| \leq \frac{n \times 5 \times 198}{2} = 2024 \times 495 = 1,001,880$. [...] Since $1,022,700 > 1,001,880$, we have a contradiction.

✓ Bounds the same edge count from below (Mantel's theorem on $\bar{G}$) and from above (degree at most 990), so each bound has a derivation. The written proof also contains a retracted false start and an arithmetic slip in the lower bound, which do not affect the contradiction.

Why: in Lean, a lower bound on one color cannot follow from an upper bound on the average. The Lean-grounded student derives a lower and an upper bound on the same quantity and compares them. Valid proofs on this problem: base 0%, Lean-grounded 25%.

Case study in thinking mode: certifying nonzero coefficients (IMO-ProofBench, PB-Advanced-026)

Problem. Let n and k be positive integers with $k < n$. Let $P(x)$ be a polynomial of degree n with real coefficients, nonzero constant term, and no repeated roots. Suppose that for any real numbers $a_0, a_1, \ldots, a_k$ such that the polynomial $a_k x^k + \cdots + a_1 x + a_0$ divides $P(x)$, the product $a_0 a_1 \ldots a_k$ is zero. Prove that $P(x)$ has a non-real root.

Problem	GPT-6-Astra	Teacher Lean-Grounded	Teacher Traj-Guided / CoT
PB-Basic-003 Find all $f : \mathbb{R} \to \mathbb{R}$ with $(b-a)f(f(a)) = af(a+f(b))$.	$P(a,a)$ gives $f(a+f(a))=0$. If $f \not\equiv 0$, some $f(f(t)) \neq 0$, and comparing $P(t,u)$, $P(t,v)$ gives $(u-v)f(f(t))=0$, so f is injective. Hence $a+f(a)=c$.	Derives the same two facts: $af(a+f(a)) = 0$ from $P(a,a)$, and $(y-x)f(f(x)) = 0$ whenever $f(x)=f(y)$, so f is injective unless $f \circ f \equiv 0$. Keeps $c=f(0)$ free and finds the family $-x+c$.	Traj: posits $f = kx$ and cites a "known result" that $0, -x$ are the only solutions. CoT: assumes continuity. Both miss the family $-x+c$.
PB-Basic-004 Find all increasing surjective g with $g(g(x)) = g(x)+20x$.	First proves $g(0) = 0$ from the fixed point of g, then sign preservation, then studies the iterates $g^n(x)$ for $n \in \mathbb{Z}$.	First proves $g(0) = 0$: if $c = g(0) > 0$ then $g(c) > g(0) = c$, a contradiction, and similarly for $c < 0$. Then derives sign preservation before dividing by x.	Both note that g is bijective and immediately posit $g(x) = kx + c$. Traj assumes continuity for uniqueness.
PB-Basic-002 Show $\sum x^2 \geq xyzt$ given $2\sum x \geq xyzt$.	Splits on $s \geq 8$ and $s \leq 8$, using $p \leq 2s$ in the first case and $p \leq s^4/256$ in the second.	Same split and same bounds.	Chain two bounds in the same direction into $P \leq 16$ or $S \leq 8$

Table 7: **GPT-6-Astra follows the disciplines of Definition 1.** On each problem, Astra derives the intermediate facts from the hypothesis, establishes side conditions before using them, and applies each bound only in its own case. The Lean-grounded student reproduces these steps, while Traj-Guided and CoT students replace them with an ansatz, an assumed regularity, or a chain of bounds in the same direction.

Base model (thinking mode)

Let's consider the polynomial $Q(x) = (x-r_1)(x-r_2)\cdots(x-r_k)$, where $r_1, \ldots, r_k$ are the first k roots of $P(x)$. [...] This coefficient is a sum of products of $k-1$ roots, and since the roots are real and distinct, this sum is non-zero unless all such combinations sum to zero. [...] Therefore, at least one of the coefficients $a_1, \ldots, a_{k-1}$ must be non-zero, which implies that the product $a_0 a_1 \cdots a_k \neq 0$.

✗ An arbitrary choice of k real roots can give a zero coefficient ($(x-1)(x+1) = x^2 - 1$), and one nonzero coefficient does not make the product nonzero.

Lean-Grounded (thinking mode)

We claim that for any set of distinct real numbers $r_1, \ldots, r_n$, there exists a subset I of size k such that all elementary symmetric polynomials $e_0(I), e_1(I), \ldots, e_k(I)$ are nonzero. [...] 1. $P(x)$ has at least k positive roots. Choose I to be any k of these positive roots. Then every product of a subset of I consists of positive numbers, so $e_j(I) > 0$ for all j. [...] 2. $P(x)$ has at least k negative roots. Choose I to be any k of these negative roots. The product of any subset of size j has sign $(-1)^j$. Thus, $e_j(I) \neq 0$ for all j [...] 3. $P(x)$ has fewer than k positive and fewer than k negative roots. [...] it is a known combinatorial fact that one can always find k numbers whose elementary symmetric polynomials are all nonzero.

∼ Cases 1 and 2 are proved: roots of the same sign certify that every coefficient is nonzero. Case 3 is asserted without proof.

Why: a Lean proof cannot claim that a coefficient is nonzero without a certificate. The Lean-grounded student supplies one (a sign argument) and isolates the case where it does not apply, while the base model relies on a false general claim.

A.2.4 Proof Patterns for Frontier LLMs

Table 7 compares GPT-6-Astra with our students on three problems. Astra follows the disciplines of Definition 1 on each of them, and the Lean-grounded student follows the same route. For derivation discipline, on PB-Basic-003 both derive $f(a+f(a))=0$ from $P(a,a)$ and show that f is injective unless $f \circ f \equiv 0$, while Traj-Guided and CoT students posit $f = kx$ or assume continuity and miss the family $-x+c$. For dependency discipline, on PB-Basic-004 both prove $g(0)=0$ and sign preservation before dividing by x, while the other students posit $g(x) = kx+c$ directly. For closure, on PB-Basic-002 both split on $S \leq 8$ and $S > 8$ and apply each bound only in its own case, while the other students chain two bounds in the same direction into a false claim.

A.2.5 TEST-TIME LEAN-ASSISTED METHOD LACKS ROBUSTNESS IN INFORMAL–FORMAL ALIGNMENT

Case study: test-time Lean verification vs. Lean-grounded training (AIME 2025 I, Problem 3)

Problem. The 9 members of a baseball team went to an ice cream parlor after their game. Each player had a single-scoop cone of chocolate, vanilla, or strawberry ice cream. At least one player chose each flavor, and the number of players who chose chocolate was greater than the number of players who chose vanilla, which was greater than the number of players who chose strawberry. Let N be the number of different assignments of flavors to players that meet these conditions. Find the remainder when N is divided by 1000. **Answer:** 16.

HERMES with Qwen3-8B (16 Lean checks, 2,649 s, gives up)

Step 1. The number of valid triples (c, v, s) of positive integers such that $c + v + s = 9$, $c > v > s$, and $c, v, s \geq 1$ is equal to the number of integer partitions of 9 into 3 distinct positive parts [...] **Lean check:** passes by `rfl`

```
theorem test :
  let valid_triples := Finset.filter (fun triple : ℕ × ℕ × ℕ =>
    let (c, v, s) := triple
    c + v + s = 9 ∧ c > v ∧ v > s ∧ c ≥ 1 ∧ v ≥ 1 ∧ s ≥ 1)
    (Finset.range 10 ×ˢ Finset.range 10 ×ˢ Finset.range 10)
  let valid_partitions := Finset.filter (fun partition : ℕ × ℕ × ℕ =>
    let (c, v, s) := partition
    c + v + s = 9 ∧ c > v ∧ v > s ∧ c ≥ 1 ∧ v ≥ 1 ∧ s ≥ 1)
    (Finset.range 10 ×ˢ Finset.range 10 ×ˢ Finset.range 10)
  valid_triples.card = valid_partitions.card := by
  intro valid_triples valid_partitions
  have h_main : valid_triples.card = valid_partitions.card := by
    rfl
  exact h_main
```

Step 9. The number of integer partitions of 9 into 3 distinct positive parts is equal to the number of integer partitions of 9 into 3 distinct positive parts where the parts are in strictly decreasing order. [...] **Lean check:** fails, proof left as `sorry`

```
theorem test :
  let partitions_9_3_distinct := {s : Finset ℕ | s.card = 3 ∧ s.sum id = 9 ∧ ∀ x ∈
    s, x > 0}
  let partitions_9_3_distinct_decreasing := {s : Finset ℕ | s.card = 3 ∧ s.sum id = 9
    ∧ ∀ x ∈ s, x > 0 ∧
    ∃ a b c, s = {a, b, c} ∧ a > b ∧ b > c}
  Set.ncard partitions_9_3_distinct = Set.ncard partitions_9_3_distinct_decreasing :=
    by sorry
```

Steps 10–15. Six further restatements of the same equivalence. None of them passes.

✗ In step 1, the two sets are defined by the same filter, so the check passes by `rfl` while the correspondence between triples and partitions is never formalized. When that correspondence is stated in step 9, the prover cannot prove it. The loop then restates the same claim until it gives up, before any count is made. Counting partitions would also miss the multinomial factor for assigning flavors to players.

Lean-grounded Qwen3-8B (no Lean at inference, 15 of 16 samples correct)

From $c + v + s = 9$ and $c > v > s$: The smallest values for c and v in terms of s are: $v \geq s + 1$, $c \geq v + 1 \geq s + 2$. So, the minimum sum $c + v + s$ for a given s is: $(s + 2) + (s + 1) + s = 3s + 3$. Since the total number of players is 9: $3s + 3 \leq 9$, $3s \leq 6$, $s \leq 2$. [...] So, for $s = 1$, there are 2 valid combinations for (c, v, s): $(6, 2, 1)$, $(5, 3, 1)$. [...] So, for $s = 2$, there is 1 valid combination for (c, v, s): $(4, 3, 2)$. [...] For each valid triplet (c, v, s), we need to calculate the number of ways to assign these counts to the 9 players. This is a multinomial coefficient: $\frac{9!}{c!\, v!\, s!}$. [...] $N = 252 + 504 + 1{,}260 = 2{,}016$. [...] Thus, the remainder is 16.

✓ Derives the bound $s \leq 2$, closes every case, and counts assignments with the multinomial coefficient.

Why: step-level verification spends its budget on claims that are easy to state but hard to formalize, and a check can pass on a statement that no longer carries the claim. Lean-grounded training moves the structure of a formal proof into the model itself: bound the variables, exhaust the cases, and count each case. The student then writes a complete solution in one pass, without any verifier.

Component	Model	Hardware
Reasoner	Qwen3-8B	4× NVIDIA L40S
Formalizer	Goedel-Formalizer-V2-8B	4 × NVIDIA RTX 5880
Prover	Goedel-Prover-V2-8B	8× NVIDIA H100
Lean REPL	Lean 4.17.0 + Mathlib	CPU, 32 workers, 10 GB each

Table 8: **Deployment of our HERMES replication.** Each model is served as a separate endpoint, and every step is checked in a Lean REPL.

A.3 Additional Settings

A.3.1 Settings for RLVR of Qwen3-4B-Base

We use Qwen3-4B-Base because it has undergone pretraining but no post-training, so differences between runs can be attributed to the SFT initialization instead of prior RL or OPD. We construct the SFT data from MATH-train problems and run RL on problems from OpenR1, whose final answers can be checked automatically and thus provide a rule-based reward without a learned verifier. All runs share the DAPO (Yu et al., 2025a) algorithm, ORM reward, and 4,096 response token length, training batch size 128, lr 1e-6, mini-batch size 16, rollout number 8.

A.3.2 Settings for Hermes

See Table 8.

A.4 Potential Advanced Implementations of Lean-Grounded Learning

Our implementation is deliberately simple. A single offline revision pass followed by SFT makes the controlled conditions differ only in the privileged information given to the teacher, so the gains can be attributed to ρ. Stronger objectives can use ρ more fully: 1. on-policy distillation (OPD) (Lu & Lab, 2025), where a teacher conditioned on ρ scores every token of the student's own proofs through the reverse KL $\mathrm{KL}\big(\pi_\theta(\cdot \mid x, y_{<t}) \,\|\, \pi_T(\cdot \mid x, \rho, y_{<t})\big)$, 2. process rewards for RL (Lightman et al., 2024), where a judge given ρ checks each rollout against Definition 1, and 3. iterative refinement and grounding, where problems the student fails on are autoformalized and proved to revise its drafts in the next round. We leave these directions to future work.